applied sciences

MDPI

Article

Weight Adjustment Framework for Self-Attention Sequential Recommendation

Zheng-Ang Su and Juan Zhang *

School of Electronic and Electrical Engineering, Shanghai University of Engineering Science, Shanghai 201620, China; suzhengang9205@163.com
* Correspondence: zhang-j@foxmail.com; Tel.: +86-158-2171-2631

Abstract: In recent years, sequential recommendation systems have become a hot topic in the field of recommendation system research. These systems predict future user actions or preferences by analyzing their historical interaction sequences, such as browsing history and purchase records, and then recommend items that users may be interested in. Among various sequential recommendation algorithms, those based on the Transformer model have become a focus of research due to their powerful self-attention mechanisms. However, one of the main challenges faced by sequential recommendation systems is the noise present in the input data, such as erroneous clicks and incidental browsing. This noise can disrupt the model's accurate allocation of attention weights, thereby affecting the accuracy and personalization of the recommendation results. To address this issue, we propose a novel method named "weight adjustment framework for self-attention sequential recommendation" (WAF-SR). WAF-SR mitigates the negative impact of noise on the accuracy of the attention layer weight distribution by improving the quality of the input data. Furthermore, WAF-SR enhances the model's understanding of user behavior by simulating the uncertainty of user preferences, allowing for a more precise distribution of attention weights during the training process. Finally, a series of experiments demonstrate the effectiveness of the WAF-SR in enhancing the performance of sequential recommendation systems.

Keywords: sequential recommendation; deep learning; self-attention mechanism; denoise

check for updates

Citation: Su, Z.-A.; Zhang, J. Weight Adjustment Framework for Self-Attention Sequential Recommendation. *Appl. Sci.* **2024**, *14*, 3608. https://doi.org/10.3390/app14093608

Academic Editor: Luis Javier Garcia Villalba

Received: 24 March 2024
Revised: 16 April 2024
Accepted: 22 April 2024
Published: 24 April 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

With the advent of the internet and the digital age, users are faced with a vast amount of information and choices. Whether they are online shopping, selecting movies or books, or browsing news, users find it challenging to locate content of interest among the plethora of options available. It is in this context that recommendation systems (RSs) have emerged as a key technology to address this challenge. Recommendation systems employ various recommendation algorithms to help users filter information in an information-redundant environment, providing more choices that users might be interested in. Recommendation systems also utilize advanced machine learning techniques, such as collaborative filtering (CF) [1], to analyze similarities among users. This means that the system can recommend content based on the behavior of other users with similar preferences.

In traditional recommendation systems [2], although user preferences and behaviors are considered, these methods often overlook the temporal sequence and dynamic nature of user behavior. Sequential recommendation (SR) is a significant branch within the recommendation system field [3], focusing on leveraging the time-sequenced information of user behaviors to generate personalized recommendations. Unlike traditional recommendation systems, sequential recommendation systems specifically highlight the order and dynamic evolution of user behaviors to offer more precise and relevant content or product recommendations. By taking into account the sequence of users' historical interactions, sequential recommendation is capable of predicting the items that users might interact with next.

Traditional methods in sequential recommendation primarily include techniques such as Markov chains, matrix factorization [4], and k-nearest neighbor (KNN) [5]. These methods played a pivotal role in the early development of sequential recommendation. Rendle et al. [6] introduced the factorizing personalized Markov chains (FPMC) model. The core idea of FPMC is to combine traditional Markov chain approaches with matrix factorization techniques, constructing a personalized transition matrix for each user to form a transition cube. This approach retains the sequence-modeling capabilities of Markov chains while incorporating considerations for individual user preferences. In sequential recommendation, KNN is a method that considers the sequence of user behaviors [7]. The core idea of KNN is to find the K most similar neighbors to the target user and predict the target user's preferences for unknown items based on the behaviors or ratings of these neighbors or to recommend items to them. These traditional methods have achieved certain results in the field of recommendation systems, but their limitations in processing complex sequence data and understanding deep user preferences have spurred on the development of sequential recommendation methods based on deep learning.

In recent years, deep learning technology has made significant progress with regard to recommendation algorithms [8–10]. With the successful application of deep learning technology in fields such as image recognition and natural language processing, researchers have begun to apply deep learning models to recommendation systems, especially in the field of sequential recommendation. Zhou et al. [11] proposed the deep interest network (DIN), specifically designed to improve the accuracy of click-through rate predictions. The core of this model lies in its unique user-interest-modeling approach, which captures the dynamic changes in user interests by learning deep representations of users' historical behaviors.

With the rise of the Transformer architecture [12], Transformer-based sequential recommendation methods have increasingly garnered attention, such as SASRec [13] and BERT4Rec [14]. The Transformer utilizes the self-attention mechanism to effectively model long-sequence data, address long-range dependency issues, and boasts an efficient computational speed. However, recent research [15] has revealed that the attention mechanism might suffer from inaccuracies in weight allocation, with noisy data inputs [16,17] and suboptimal position encoding [18] identified as primary causes. The existing sequential recommendation models typically assume that the target item is related to all historical interaction items. However, in real-world scenarios, this assumption may not always hold true, as users may inadvertently interact with items that deviate from their interests or preferences, leading to noise in the interaction data. These noise sources are difficult to identify, which presents challenges in accurately discerning users' true preferences through the self-attention mechanism, resulting in inaccurate attention weight allocation.

Inspired by previous works [15,19,20], we address the issue of inaccurate attention weight distribution from two perspectives. (1) We introduce a denoising technique to mitigate the influence of noisy data input on the attention weight distribution. We set up a filter layer to clean the data before it flowed into the self-attention layer, to prevent noise from negatively affecting the model's interpretation of user behavior and the accuracy of its recommendations. (2) We enhanced the model's capability to understand user behavior, thereby refining the attention weight distribution. We enhanced the model's grasp of the logic behind user behavior by introducing a logical representation layer. This logical representation layer enabled the attention mechanism to more effectively identify the user interactions that had the greatest impact on the recommendation outcomes.

In summary, our contributions are as follows:

- We propose a novel sequential recommendation method, WAF-SR, which enhances the accuracy of attention weight allocation by mitigating the interference of noise in the input data;
- We introduce logical representation technology to correct and optimize the allocation of weights by the attention mechanism more precisely;

- We have conducted extensive experiments on three public datasets, and the results validate the effectiveness and superiority of our proposed method.

The structure of this paper is organized as follows: Section 1 serves as the introduction, presenting the research background, objectives, and significance of the study. Section 2 provides a review of previous work closely related to our research. Section 3 elaborates, in detail, our research methodology, including the formal definition of the problem and key components: the filter layer, self-attention layer, logical representation layer, and prediction layer. Section 4 describes the specific setup and steps of the experiments, covering the selection of datasets, evaluation metrics, choice of baseline models, technical details of implementation, overall performance of the model, ablation studies, and research on hyperparameters. Section 5 summarizes the findings of this study and includes the discussion of the results.

2. Related Work

In the early stages of recommender systems, Rendle et al. [21] proposed a universal optimization criterion for personalized ranking called BPR, which played a revolutionary role in the research of recommender systems. BPR aims to directly optimize the quality of rankings, rather than the traditional accuracy of rating prediction. Through this method, BPR can more effectively handle implicit feedback data, such as clicks or purchases. With the advancement of deep learning technology, an increasing number of people are applying deep learning techniques to the field of sequential recommendation. Researchers have started to use complex network models such as convolutional neural networks (CNNs), recurrent neural networks (RNNs), and graph neural networks (GNNs).

GRU4Rec [22] was the first to utilize RNNs to model the sequence of user behaviors within a session to predict the user's next possible action. The session-modeling part employs RNNs to capture the sequence of user behaviors within a session, generating a fixed-length vector representation. The prediction part takes this vector representation as input and uses a feedforward neural network to predict the next possible action. Caser [23] models user interests by embedding items in a sequence into a continuous vector space and then applying CNNs to the embedded vectors to extract significant features within the item sequence. Wu et al. [24] proposed a graph-neural-network-based session recommendation model, SR-GNN, aimed at exploring complex transition relationships between items and generating accurate item-embedding vectors. SR-GNN constructs directed graphs from sequences and captures transitions between items through this structure, thereby producing precise item-embedding vectors.

Recently, Transformer-based methods for sequential recommendation have emerged in the mainstream, due to their efficient parallel computation capabilities and their significant potential in capturing the sequential behaviors of users. SASRec [13] replaces RNNs with the self-attention mechanism to better capture long-term dependencies and global relationships, efficiently handling long sequences. Although SASRec effectively encodes user historical interactions into hidden vectors for recommendation using a self-attention mechanism, the inherent limitations of its unidirectional model impact the completeness of the hidden representations in the user-behavior sequence. To address this issue, BERT4Rec [14] employs a deep bidirectional self-attention mechanism to more comprehensively model the user-behavior sequence. This allows each historical item in the sequence to consider the contextual information from both its left and right sides for more accurate recommendations. Similar to the original Transformer model, SASRec is fundamentally a non-personalized model that does not include personalized embeddings for individual users. SSE-PT [25] overcomes the non-personalization limitations of the SASRec model by introducing stochastic shared embedding (SSE) regularization.

SASRec and BERT4Rec, which are recommendation systems based on self-attention networks (SANs), exhibit three main limitations. Firstly, traditional sequential recommendation systems must handle a large number of items, but due to the long-tail distribution of items, many have sparse interaction data. This leads to the insufficient training of em-

beddings for these infrequently interacted-with items, thereby affecting the accuracy of the corresponding attention weights. Secondly, in the SAN model, the item embeddings and position embeddings are directly added together. This method may introduce noise and limit the model's ability to capture patterns in user-behavior sequences. Thirdly, the SAN model requires direct interactions between all historical items of a user (termed item-to-item interactions), which results in the computational requirements in terms of time and space growing quadratically as the length of the historical sequence increases. This can make the actual operational costs prohibitively high.

To address the high operational costs and the noise issues caused by the direct addition of item embedding vectors and positional embedding vectors, LightSAN [18] introduces a low-rank decomposed self-attention mechanism. This adjustment means that the time and space complexity of the SAN is linearly related to the length of the user's historical records. Additionally, by independently calculating the positional relevance, LightSAN effectively eliminates noisy correlations, thereby better modeling the sequential patterns in user behavior. LightSAN cannot explicitly utilize low-level spatial information, including sequence and distance information. Spatial information helps enhance the representational capability of positional encoding in error-prone data. AC-TSR [15] improves upon the limitations of LightSAN by explicitly utilizing spatial relationships to compute attention weights with greater structural significance. We have proposed a method called WAF-SR, based on AC-TSR. WAF-SR has improved the processing of input noise data. At the same time, we have added a logical representation layer to enhance the model's understanding of user-interaction patterns.

Additionally, Locker [26] and TiSASRec [27] have, respectively, improved the self-attention mechanism for modeling short-term user behaviors and the issue of implicit time modeling. Locker [26] enhances the model's ability to capture short-term user dynamics effectively by integrating local encoders with existing global attention heads. TiSASRec [27] explores the impact of different time intervals on the prediction of the next item by explicitly modeling the interaction timestamps in the sequence and the time intervals between interactions.

3. Methodology

This section introduces our proposed method, WAF-SR. WAF-SR consists of a filter layer, a self-attention layer, a logical representation layer, and a prediction layer. We have adopted two key strategies to address the issue of inaccurate attention weight distribution. (1) We have introduced denoising techniques to mitigate the impact of noise data. We have integrated a filter layer to denoise user-interaction data before it enters the self-attention layer. (2) We have enhanced the model's understanding of user behavior by introducing a logical presentation layer. Finally, the prediction layer makes the final recommendation decisions. In the following sections, we will further elaborate on the specific computational methods of each WAF-SR component and their roles within the overall framework. Our framework is illustrated in Figure 1.

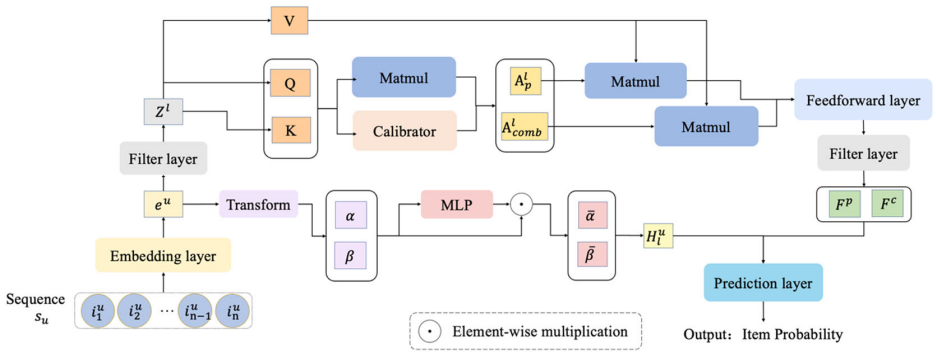

Figure 1. Overview of the proposed WAF-SR framework.

3.1. Problem Formulation

In this section, we will provide the problem definition for sequential recommendation. Assuming that we had a series of user–item historical interaction sequences, our goal was to predict the item that a user was most likely to be interested in next, based on these historical sequences. Define U as the set of users, and I as the set of items. At each time step, the user u $\in$ U interacts with an item from the set I, forming a sequence $S_u = \left\{ i_1^u, \ldots, i_{|S_u|}^u \right\}$, where $i_k^u \in I (1 \leq k \leq |S_u|)$ represents the item u interacted with at the k-th time step and $|S_u|$ is the length of the sequence for user u. Our model attempts to learn patterns from the sequence S_u and use these patterns to predict the item that user u is most likely to choose next. Mathematically, we can represent this prediction task as an optimization problem, where we seek a probability model P to maximize the likelihood of the next item $i_{|S_u|+1}^u$ being a specific item i, given the known sequence S_u. This can be formulated as follows:

$$\underset{i \in I}{\operatorname{argmax}} P\left(i_{|S_u|+1}^u = i \mid S_u \right) \tag{1}$$

Furthermore, to capture and express the characteristics of each item and the user's interactions with them more accurately, we introduced an embedding matrix to map the IDs of the items to a continuous vector space. Create an item-embedding matrix $M \in \mathbb{R}^{|I| \times d}$, where d is the dimension of each vector and $|I|$ represents the number of all items. Given the input sequence S_u, embedding this sequence, S_u, results in $\mathrm{e}^u \in \mathbb{R}^{n \times d}$, where $e^u = \{m_{s_1}, m_{s_2}, \ldots, m_{s_n}\}$; here, $m_{s_k} \in \mathbb{R}^d$ represents the embedding of the item at position k in the sequence and n represents the length of the sequence.

3.2. Filter Layer

The filter layer is a denoising module. User-interaction data recorded on online platforms inevitably contain noise. Previous studies, such as SSE-PT [25], TiSASRec [27], and AC-TSR [15], although demonstrating significant recommendation effectiveness in sequential recommendation systems, did not systematically consider the handling of input data noise in their model designs. This oversight may limit their robustness in practical application scenarios. To solve the problem of the inaccurate allocation of attention weights due to noise input, we added a filter layer to WAF-SR. The filter layer serves to identify and remove noise or irrelevant interaction data from the input user-interaction sequences. This layer utilizes signal processing techniques to preprocess the input sequences, ensuring that the data passed to the self-attention layer are purified. Specifically, the embedding matrix of the item e^u is first transformed into the frequency domain via the fast Fourier transform (FFT) [28,29]:

$$x^l = \mathcal{F}(e^u) \in \mathbb{C}^{n \times d} \tag{2}$$

where $\mathcal{F}(\cdot)$ denotes the one-dimensional fast Fourier transform, x^l is a complex tensor representing the spectrum of e^u, $\mathbb{C}$ represents the complex field, and $\mathbb{C}^{n \times d}$ represents a complex matrix with n rows and d columns. The spectrum is modulated by multiplying it with a learnable filter $W \in \mathbb{C}^{n \times d}$:

$$\tilde{x^l} = W \bigodot x^l \tag{3}$$

where $\odot$ represents element-wise multiplication. Then, modulate the spectrum $\tilde{x^l}$ back into the time domain through the inverse fast Fourier transform to update the sequence representation:

$$Z^l = \mathrm{LayerNorm}\left(e^u + \mathrm{Dropout}\left(\mathcal{F}^{-1}\left(\tilde{x^l} \right) \right) \right) \tag{4}$$

where $\mathcal{F}^{-1}(\cdot)$ stands for the one-dimensional inverse fast Fourier transform, which converts the complex tensor into a real-valued tensor. To prevent gradient vanishing, skip connections [30], layer normalization [31], and dropout [32] operations are also introduced

in the filter layer. It can be summarized that the item's embedding matrix e^u passes through the filter layer to obtain the filtered matrix Z^l:

$$Z^l = FL(e^u) \tag{5}$$

where $l \in 0, 1, 2, \ldots, n$ represents the number of layers of the filter layer, FL denotes the filter layer. When $l = 0$, $Z^l = e^u$.

3.3. Self-Attention Layer

The self-attention layer is the core of the sequential recommendation system, calculating the relationship weights between elements within the sequence through the self-attention mechanism. This layer enables the model to highlight the most critical interaction features and suppress less important information, optimizing the distribution of attention weights. This is particularly crucial for capturing the temporal dependencies in user behavior. This section will introduce the computational methods of the self-attention layer.

Using the filtered matrix Z^l, calculate Q (Query), K (Key), and V (Value), respectively. As with the previous transformer, here, Q, K, and V are all obtained through linear transformations:

$$\begin{cases} Q = Z^l W^Q \\ K = Z^l W^K \\ V = Z^l W^V \end{cases} \tag{6}$$

where W^Q, W^K, and $W^V \in \mathbb{R}^{d \times d}$ are the learnable matrices in the linear transformations and d represents the dimension of the vector for each item.

To address the issue of suboptimal position encoding potentially leading to the inaccurate allocation of attention weights, we adopted a spatial calibrator scheme [15] and abandoned traditional positional encoding techniques. Specifically, first calculate the order and logarithmic distance between item pairs relative to their positions in the input sequence. We can define the actual order o_{ij} and the actual logarithmic distance d_{ij} between positions i and j in the input sequence as follows:

$$o_{ij} = \mathbb{I}(i < j) = \begin{cases} 1, & i < j \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

$$d_{ij} = \ln(1 + |i - j|) \tag{8}$$

Then, in each self-attention layer, use the query q_i^l and key k_j^l to predict the order and distance that items should have. The predicted order $\widehat{o}_{ij}$ and distance $\widehat{d}_{ij}$ can be represented as follows:

$$\widehat{o}_{ij} = \text{sigmoid}\left(affine^{(o)}\left(\left[q_i^l; k_j^l\right]\right)\right) \tag{9}$$

$$\widehat{d}_{ij} = \text{affine}^{(d)}\left(\left[q_i^l; k_j^l\right]\right) \tag{10}$$

The difference between the predictions and the true values is measured using sigmoid cross-entropy and L_2 loss, after which the original attention weights are modified:

$$s_{ij}^{(o)} = o_{ij} \ln\left(\widehat{o}_{ij}\right) + \left(1 - o_{ij}\right)\left(1 - \ln\left(\widehat{o}_{ij}\right)\right) \tag{11}$$

$$s_{ij}^{(d)} = -\frac{\theta^2\left(d_{ij} - \widehat{d}_{ij}\right)^2}{2} \tag{12}$$

$$\mathrm{A}_s = \text{softmax}\left(\frac{QK^T}{\sqrt{d}} + \mathrm{s}^{(o)} + \mathrm{s}^{(d)}\right) \tag{13}$$

where A_s represents the attention weights and θ is a learnable scalar. In the self-attention layer, the attention weights can be affected by data noise. To simulate the perturbation

process, in the l-th attention layer, we incorporate a uniform distribution μ into the attention weights A_s^l through a mask, M^l:

$$A_p^l = M^l \odot A_s^l + \left(1 - M^l\right) \odot \mu \tag{14}$$

$$M^l = \text{sigmoid}\left(\frac{Q^l W_{Q_p}^l \left(K^l W_{K_p}^l\right)^T}{\sqrt{d}}\right) \tag{15}$$

where $\odot$ denotes element-wise multiplication, Q^l and K^l are the query and key at the l-th layer, and $W_{Q_p}^l \in \mathbb{R}^{d \times d}$ and $W_{K_p}^l \in \mathbb{R}^{d \times d}$ are learnable matrices. Then, through M^l, the important parts of the weights in A_s^l are emphasized:

$$A_c^l = A_s^l \odot e^{1-M^l} \tag{16}$$

Afterwards, various attention weights are combined through a gating function:

$$A_{comb}^l = g * A_s^l + (1 - g) * A_c^l \tag{17}$$

$$g = \sigma\left(Q^l W_g^l + b_g^l\right) \tag{18}$$

where $W_g^l \in \mathbb{R}^{d \times d}$ is a trainable matrix and $b_g^l \in \mathbb{R}^d$ is also a trainable parameter. Then, the attention weights A_p^l and A_{comb}^l are fed into the feedforward layer. Note that a filter layer is added after the feedforward layer. After passing through the filtering layer, the perturbed output embeddings F^p and the calibrated output embeddings F^c are obtained.

3.4. Logical Representation Layer

Previous sequential recommendation models such as Locker [31], BERT4Rec [14], and SASRec [13] generally utilized traditional embedding representations. Traditional embedding representations often fail to fully capture the dynamic changes and uncertainties in user tastes, which may lead to suboptimal performance in recommendation systems in understanding both long-term and short-term user preferences. To address this issue, we have introduced a logical representation method to model the complexity of user behavior and preferences more accurately. In our model, the logical representation plays a crucial role; it uses a transformation matrix to map item IDs to a logical space that is better suited for representing and handling the uncertainties of user preferences. Then, multiple independent beta distributions are used to represent each item. The beta distribution is a continuous probability distribution defined within the range [0, 1], which simulates the uncertainty or variability of user preferences for items. The beta distribution is defined as follows:

$$p_{[(\alpha,\beta)]}(x) = \frac{1}{B(\alpha,\beta)} x^{\alpha-1}(1-x)^{\beta-1} \tag{19}$$

where $x \in [0,1]$, and the shape parameters $\alpha, \beta \in [0,\infty]$ determine the shape of the beta distribution, including its peak position, degree of skewness, and tail thickness. In other words, the shape parameter affects the shape and properties of the distribution, which, in turn, can be used to model uncertainty in user preferences for items. $B(\alpha,\beta)$ is the beta function. The embedding matrix of input item ID, through two transition matrices, W_α and $W_\beta \in R^{d \times d}$, transforms the item-embedding matrix e^u into two shape matrices, α and $\beta \in R^{m \times d}$, where m is the length of the user-interaction sequence:

$$\begin{cases} \alpha = e^u W_\alpha \\ \beta = e^u W_\beta \end{cases} \tag{20}$$

Define a multi-dimensional vector $v_i = [(\alpha_i, \beta_i)] = [(\alpha_{i,1}, \beta_{i,1}), (\alpha_{i,2}, \beta_{i,2}), \cdots, (\alpha_{i,d}, \beta_{i,d})]$, each dimension is described by a pair of shape parameters of the beta distribution to model the uncertainty in that dimension. Its probability density function $p_{v_i}(x)$ can be expressed as follows:

$$p(v_i) = \left[p_{[(\alpha_{i,1},\beta_{i,1})]}(x), p_{[(\alpha_{i,2},\beta_{i,2})]}(x), \cdots, p_{[(\alpha_{i,},\beta_{i,d})]}(x)\right] \tag{21}$$

This is a multidimensional expression, and each dimension has a corresponding beta distribution. Given a user behavior $\mathcal{V}_u = \{v_1, v_2, \cdots, v_m\}$, the corresponding embedding can be expressed as $v_1 = [(\alpha_1, \beta_1)], v_2 = [(\alpha_2, \beta_2)], \cdots, v_m = [(\alpha_m, \beta_m)]$. Define the output $\overline{v} = \left[(\overline{\alpha}, \overline{\beta})\right] = \mathcal{C}(\{v_1, v_2, \cdots, v_m\})$, where $\mathcal{C}$ is a probability weighting operation, which can be expressed as follows:

$$\overline{v} = \left[\left(\sum_{i=1}^{m} w_i \odot \alpha_i, \sum_{i=1}^{m} w_i \odot \beta_i\right)\right] \tag{22}$$

where Σ and $\odot$ represent the element-wise summation and product, respectively. $w_i \in R^d$ is a weight vector, and, in the j-th dimension, $w_{i,j}$ satisfies $\Sigma_{i=1}^{m} w_{i,j} = 1$. We use an attention mechanism to learn the importance of different items:

$$w_i = \frac{exp(MLP(\alpha_i \oplus \beta_i))}{\sum_j exp\left(MLP\left(\alpha_j \oplus \beta_j\right)\right)} \tag{23}$$

where $\oplus$ represents the splicing operation and *MLP* is a multi-layer perceptron that takes the connection of α and β as its input.

3.5. Prediction Layer

The prediction layer is responsible for converting the learned item representation into a specific recommendation output. Select the last elements F_n^p and F_n^c from F^p and F^c, respectively. Now, two types of item probabilities can be obtained:

$$\widehat{y}_i^P = softmax\left(F_n^p M^\top\right) \tag{24}$$

$$\widehat{y}_i^C = softmax\left(F_n^c M^\top\right) \tag{25}$$

where $M \in \mathbb{R}^{|I| \times d}$ is the item-embedding matrix. Then, calculate the perturbation loss, $\mathcal{L}_P$, and the calibration loss, $\mathcal{L}_C$:

$$\mathcal{L}_P = -\sum_{i=1}^{|\mathcal{I}|} y_i log\left(\widehat{y}_i^P\right) \tag{26}$$

$$\mathcal{L}_C = -\sum_{i=1}^{|\mathcal{I}|} y_i log\left(\widehat{y}_i^C\right) \tag{27}$$

The performance of attention affected by perturbation should be worse. Therefore, the loss can be defined as follows:

$$\mathcal{L}_{P_{final}}\left(\theta^P\right) = -\mathcal{L}_P(\theta) + \delta \mathcal{L}_{norm}\left(\theta^P\right) \tag{28}$$

$$\mathcal{L}_{norm}\left(\theta^P\right) = \sum_{l=0}^{L} \parallel 1 - m^l \parallel_2 \tag{29}$$

where θ^P and θ are parameters of the model, L is the number of layers in the transformer, and δ is a hyperparameter for balancing the loss function. $\mathcal{L}_{\text{norm}}$ is used to ensure that the perturbation is not too large and avoids the severe degradation of the model performance.

In addition, the logical representation models the variability and uncertainty of user tastes. In order to deepen the model's understanding of user preferences, we splice F_n^p, F_n^c and the logical representation H_l^u together, respectively. The logical representation H_l^u is calculated from $\overline{\alpha}$ and $\overline{\beta}$:

$$H_l^u = \frac{\overline{\alpha}}{\overline{\alpha} + \overline{\beta}} \tag{30}$$

The predicted probability $\widehat{p}^a$ and $\widehat{p}^c$ of the item can be calculated:

$$\widehat{p}^a = \left(F_n^p \oplus H_l^u\right)\left(M \oplus E\right)^\top \tag{31}$$

$$\widehat{p}^c = \left(F_n^c \oplus H_l^u\right)\left(M \oplus E\right)^\top \tag{32}$$

where $E = \frac{\alpha}{\alpha+\beta}$, uses the cross-entropy loss function to approximate the true value p:

$$\mathcal{L}_{\widehat{p}^a} = -\sum_{i=1}^{|I|} p_i log(\widehat{p}_i^a) + (1-p_i) log(1-\widehat{p}_i^a) \tag{33}$$

$$\mathcal{L}_{\widehat{p}^c} = -\sum_{i=1}^{|I|} p_i log\left(\widehat{p}_i^b\right) + (1-p_i) log\left(1-\widehat{p}_i^b\right) \tag{34}$$

Finally, the total loss of our model is as follows:

$$\mathcal{L}_{final} = \mathcal{L}_{P_{final}} + \mathcal{L}_C + \gamma\left(\mathcal{L}_{\widehat{p}^a} + \mathcal{L}_{\widehat{p}^c}\right) \tag{35}$$

where γ is a hyperparameter that balances various loss functions.

4. Experiments

4.1. Dataset

The experiment used three public datasets that are widely used in real-life scenarios: namely, Amazon Beauty, Amazon Toys, and Yelp. The statistics of the dataset are shown in Table 1.

Table 1. Statistics of the dataset.

Statistics	Users	Items	Inters	Sparsity
Beauty	22,363	12,101	198,502	99.93%
Toys	19,412	11,924	167,597	99.93%
Yelp	30,499	20,068	317,182	99.95%

The Amazon Beauty and Toys dataset consists of product reviews and metadata from Amazon, providing insights into specific product categories on Amazon, such as beauty products and toys. The Amazon Beauty dataset includes user reviews and ratings of beauty products, such as cosmetics, skincare products, etc. The Amazon Toys dataset covers user reviews and ratings of toys and game products.

Yelp is a business recommendation dataset that contains a large amount of business information, user reviews, user information, and interaction data between businesses and users. This dataset is widely used in research and experiments in the field of recommender systems.

4.2. Evaluation Metrics

The experiment selects top-k normalized discounted cumulative gain (NDCG@K) and top-k recall (Recall@K) as evaluation indicators. The value of k is selected from {10, 20}. Following previous work, we evaluate model performance in a complete ranking manner. The final ranking results are based on the entire set of items. The experiment adopts the leave-one-out method, and the last two items of the interaction sequence of each user and item are used as the verification set and test set.

4.3. Baseline Models

We select the following models as baseline models for comparison. (1) General recommendation methods: PopRec, BPR [21], and GRU4Rec [22]. (2) Sequential recommendation methods: LightSANs [26], Locker [31], BERT4Rec [14], SASRec [13], SSE-PT [25], TiSASRec [27], and AC-TSR [15]. Note that the AC-TSR that we compare is implemented based on SASRec.

4.4. Implementation Details

In the experiments, both the baseline models and our proposed method's implementation leveraged the RecBole framework [33], an open-source, deep-learning-based recommendation system framework. For all baseline models and our proposed method, Adam was used as the optimizer for training over 200 epochs, with the batch size set to 256. The sequence length was set to 50, and the learning rate was 1×10^{-4}. The hyperparameter γ in Equation (35) was set at 0.003 for all datasets. We employed a grid-search strategy to configure the optimal parameter settings for each dataset, where the hidden size was selected from {64, 128}, the inner size from {64, 128}, the number of self-attention layers from {2, 3, 4}, and the number of attention heads from {2, 4, 8}. Notably, for the Beauty dataset, we configured the self-attention mechanism with three layers and eight heads. For the Toys dataset, the configuration comprised four self-attention layers and two heads. As for the Yelp dataset, we utilized three self-attention layers and four heads. The hidden size was set at 128 for both the Toys and Yelp datasets and at 64 for the Beauty dataset. Additionally, the inner size was assigned as 64 for the Toys and Yelp datasets and as 128 for the Beauty dataset. All computational tasks were efficiently performed on an NVIDIA GeForce RTX 2080 GPU, which is manufactured by NVIDIA Corporation, headquartered in Santa Clara, CA, USA. Our development and experimental setup were conducted in a Linux environment, with code constructed in Python and PyCharm used as the integrated development environment (IDE), supporting the coding, debugging, and management of the project.

4.5. Overall Performance

The comparison of our method with various baseline models is shown in Table 2. From Table 2, it can be seen that SASRec significantly outperforms earlier methods, such as PopRec, BPR, and GRU4Rec. Models based on the transformer architecture demonstrate a notable advantage in modeling long-sequence data. This advantage is primarily due to the transformer's ability to capture dependencies across longer sequences more effectively than traditional methods, which is crucial for understanding user-behavior patterns and improving recommendation accuracy. Furthermore, the use of self-attention mechanisms allows these models to focus on the most relevant parts of the input sequences, enhancing their predictive performance. Additionally, side information can have a certain impact on the performance of sequential recommendation systems. Methods that integrate side information, such as TiSASRec, outperform the basic SASRec model. The inclusion of side information allows the model to gain a deeper understanding of user preferences. However, incorporating too much side information might increase the complexity of the model. The superior performance of AC-TSR is attributed to its consideration of noisy inputs on attention weights. We improved the model's denoising ability based on AC-TSR and modeled the uncertainty of user preferences, showing the advanced nature of our proposed method across multiple datasets.

Table 2. Overall performance. Bold data represent the best results. Underlined data signify the second-best results.

Model	Yelp				Toys				Beauty			
	Recall		NDCG		Recall		NDCG		Recall		NDCG	
	@10	@20	@10	@20	@10	@20	@10	@20	@10	@20	@10	@20
PopRec	0.0099	0.0161	0.0051	0.0067	0.0105	0.0172	0.0060	0.0077	0.0157	0.0242	0.0076	0.0097
BPR	0.0589	0.0830	0.0324	0.0384	0.0344	0.0560	0.0151	0.0205	0.0375	0.0590	0.0168	0.0222
GRU4Rec	0.0418	0.0679	0.0206	0.0271	0.0449	0.0708	0.0221	0.0287	0.0654	0.1002	0.0322	0.0410
LightSANs	0.0630	0.0904	0.0385	0.0453	0.0768	0.1116	0.0354	0.0442	0.0770	0.1177	0.0358	0.0461
Locker	0.0603	0.0869	0.0380	0.0446	0.0755	0.1094	0.0345	0.0430	0.0802	0.1197	0.0365	<u>0.0464</u>
SASRec	0.0618	0.0879	0.0387	0.0453	0.0776	0.1100	0.0352	0.0434	0.0779	0.1152	0.0353	0.0447
BERT4Rec	0.0467	0.0710	0.0264	0.0325	0.0489	0.0769	0.0253	0.0324	0.0557	0.0868	0.0279	0.0358
SSE-PT	0.0556	0.0779	0.0323	0.0379	0.0560	0.0837	0.0255	0.0325	0.0587	0.0936	0.0278	0.0366
TiSASRec	0.0618	0.0909	0.0387	0.0460	0.0819	<u>0.1171</u>	0.0367	0.0456	0.0794	0.1208	0.0356	0.0461
AC-TSR	<u>0.0664</u>	<u>0.0955</u>	<u>0.0407</u>	<u>0.0480</u>	<u>0.0825</u>	0.1166	<u>0.0371</u>	<u>0.0456</u>	<u>0.0817</u>	<u>0.1218</u>	<u>0.0375</u>	0.0454
WAF-SR	**0.0688**	**0.1014**	**0.0419**	**0.0501**	**0.0849**	**0.1219**	**0.0382**	**0.0474**	**0.0844**	**0.1258**	**0.0389**	**0.0493**

4.6. Ablation Study

We conducted ablation studies on the Beauty dataset, and the experimental results are shown in Table 3. In Table 3, WAF-SR represents the performance without removing any modules. We performed eliminations on the logic representation layer and the filter layer, respectively. When only the logic representation layer is eliminated, Recall@10 and Recall@20 decrease by 0.0062 and 0.0082, respectively. NDCG@10 and NDCG@20 decrease by 0.003 and 0.0034, respectively. When only the filter layer is eliminated, Recall@10 and Recall@20 decrease by 0.0037 and 0.0086, respectively. NDCG@10 and NDCG@20 decrease by 0.0026 and 0.0038, respectively. The experiments demonstrate that both the logic representation layer and the filter layer significantly enhance the model. WAF-SR optimizes the allocation of attention weights with the action of both modules.

Table 3. Ablation study on Beauty dataset.

Model	Recall		NDCG	
	@10	@20	@10	@20
WAF-SR	0.0844	0.1258	0.0389	0.0493
w/o logical representation	0.0782	0.1176	0.0359	0.0459
w/o filter layer	0.0807	0.1172	0.0363	0.0455

4.7. Hyperparameter Study

The number of layers and heads in self-attention significantly impacts the model. We analyzed the effect of parameter settings on WAF-SR using the Toys dataset. As shown in Table 4, we fixed the number of heads at two, and as the number of layers increased to the third layer, the gains began to diminish, possibly due to overfitting leading to a decline in model performance. Due to limited experimental equipment resources, we did not stack more layers. We visualized the impact of the attention layer number, as shown in Figure 2. With the number of layers fixed at two, as the number of heads increased (starting from two heads, with performance data identical to when the number of layers was two), the overall performance of the model did not improve. Too many heads might introduce too many parameters, increasing the model's complexity and leading to overfitting or a decrease in learning efficiency. Therefore, when designing models based on self-attention, it is crucial to choose an appropriate number of layers and heads to ensure that the model can maintain sufficient expressive power while avoiding unnecessary computational resource wastage and performance degradation. Through further experimental analysis, we recommend carefully balancing different combinations of layers and heads when applying the self-attention mechanism to find the optimal configuration balance. This balance helps improve model performance while controlling model complexity and computational costs.

Table 4. The impact of different numbers of layers and heads in the attention layer of the Toys dataset.

Settings	Recall		NDCG	
	@10	@20	@10	@20
1 layer	0.0846	0.1205	0.0376	0.0466
2 layers	0.0849	0.1219	0.0382	0.0474
3 layers	0.0845	0.1223	0.0375	0.0470
4 heads	0.0834	0.1229	0.0371	0.0471
8 heads	0.0839	0.1225	0.0376	0.0473

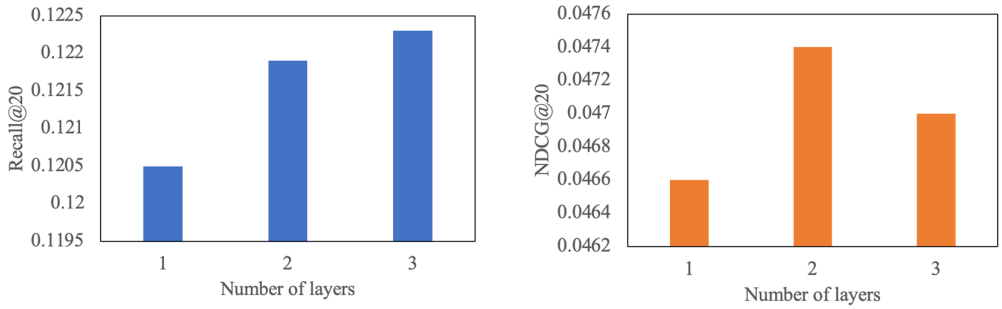

Figure 2. The influence of the number of self-attention layers. (**a**) The influence of the number of self-attention layers on Recall@20. (**b**) The influence of the number of self-attention layers on NDCG@20.

We analyzed the impact of different values of the hyperparameter δ in Equation (28) on the Beauty dataset, and the experimental results are shown in Figure 3. On the Recall@20 metric, the model achieves a superior performance when the hyperparameter δ is set to either 0.03 or 0.07. Furthermore, the model obtains an outstanding performance on the NDCG@20 metric at a δ value of 0.07. By appropriately adjusting the value of δ, we can effectively enhance the model's performance in terms of accuracy and personalized recommendations. Considering the performance of the above two indicators, we ultimately set the value of δ to 0.07 for the Beauty dataset and to 0.03 for both the Toys and Yelp datasets.

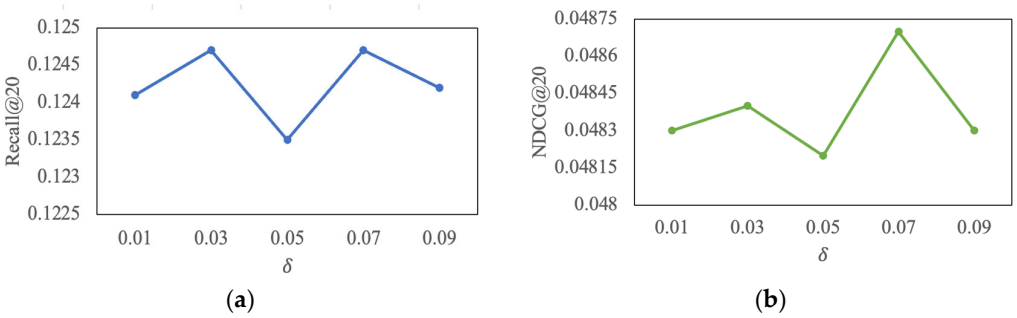

Figure 3. The impact of different values of the hyperparameter δ. (**a**) The impact of different values of hyperparameter δ on Recall@20. (**b**) The impact of different values of hyperparameter δ on NDCG@20.

5. Conclusions

Self-attention-based sequential recommendation models currently face challenges due to inaccurate attention weight distributions caused by noisy data inputs. We addressed this challenge with two approaches, building on the AC-TSR [15] model. AC-TSR reallocates attention weights based on the contribution of each historical item to the model's predictions to mitigate the issue of noisy inputs. Unlike AC-TSR, we added a denoising module and a logical representation layer. The denoising module enhances data quality, thereby improving the distribution of attention layer weights. Additionally, the logical representation layer models the uncertainty of user preferences and deepens the model's understanding of user behavior, thus refining the attention weight distribution during training. Ultimately, we validated the effectiveness of our proposed method through extensive experiments.

We compared WAF-SR with several baseline methods, and, across multiple datasets and various evaluation metrics, WAF-SR consistently achieved the highest accuracy. We conducted ablation studies to individually remove the filtering and logical representation layers, demonstrating that these added components significantly impact recommendation performance. We also studied the hyperparameters within the model, observing the impact on WAF-SR by adjusting the values of these parameters. We provided settings for some of WAF-SR's parameters to enable researchers to replicate our experiments. In future production implementations, not only does the WAF-SR algorithm have the potential to

help enterprises uncover market trends and user needs, providing robust data support for product development and marketing strategy adjustments, but its superior recommendation performance can also significantly enhance user satisfaction and conversion rates. However, despite its considerable advantages, WAF-SR also poses certain limitations. Specifically, the introduction of the denoising module and logical representation layer may lead to model overfitting, especially in scenarios with small datasets or limited feature diversity. Given these challenges, we plan to further research and develop more efficient and robust denoising techniques in future work to reduce the likelihood of overfitting and enhance the model's generalizability across various application scenarios.

Author Contributions: Z.-A.S. performed experiments, contributed to the design of the sequential recommendation model, performed detailed analyses, and wrote the manuscript. J.Z. set the direction of the study and made final corrections. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original data presented in the study are openly available in Amazon Review Data at http://jmcauley.ucsd.edu/data/amazon/, accessed on 20 March 2024, and Yelp Open Dataset at https://www.yelp.com/dataset, accessed on 20 March 2024. Our source code is available at https://github.com/AngSZ/WAF-SR/tree/main, accessed on 20 March 2024.

Acknowledgments: The equipment and environment required for the experiment were generously provided by Shanghai University of Engineering Science, for which we express our sincere gratitude for the substantial support from the institution.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Sarwar, B.; Karypis, G.; Konstan, J.; Riedl, J. Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th International Conference on World Wide Web, Hong Kong, China, 1–5 May 2001; pp. 285–295. [CrossRef]
2. Rendle, S. Factorization Machines. In Proceedings of the 2010 IEEE International Conference on Data Mining, Sydney, Australia, 13–17 December 2010; pp. 995–1000.
3. Wang, S.; Hu, L.; Wang, Y.; Cao, L.; Sheng, Q.Z.; Orgun, M. Sequential recommender systems: Challenges, progress and prospects. *arXiv* **2019**, arXiv:2001.04830.
4. Koren, Y. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, NV, USA, 24–27 August 2008; pp. 426–434. [CrossRef]
5. Cover, T.; Hart, P. Nearest neighbor pattern classification. *IEEE Trans. Inf. Theory* **1967**, *13*, 21–27. [CrossRef]
6. Rendle, S.; Freudenthaler, C.; Schmidt-Thieme, L. Factorizing personalized markov chains for next-basket recommendation. In Proceedings of the 19th International Conference on World Wide Web, Raleigh, NC, USA, 26–30 April 2010; pp. 811–820. [CrossRef]
7. Jannach, D.; Ludewig, M. When recurrent neural networks meet the neighborhood for session-based recommendation. In Proceedings of the Eleventh ACM Conference on Recommender Systems, Como, Italy, 27–31 August 2017; pp. 306–310.
8. Ma, J.; Sun, T.; Zhang, X. Time Highlighted Multi-Interest Network for Sequential Recommendation. *Comput. Mater. Contin.* **2023**, *76*, 3569–3584. [CrossRef]
9. Li, Z.; Sun, A.; Li, C. DiffuRec: A Diffusion Model for Sequential Recommendation. *arXiv* **2023**, arXiv:2304.00686. [CrossRef]
10. Yue, Z.; Wang, Y.; He, Z.; Zeng, H.; McAuley, J.; Wang, D. Linear Recurrent Units for Sequential Recommendation. *arXiv* **2023**, arXiv:2310.02367.
11. Zhou, G.; Zhu, X.; Song, C.; Fan, Y.; Zhu, H.; Ma, X.; Yan, Y.; Jin, J.; Li, H.; Gai, K. Deep interest network for click-through rate prediction. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, 19–23 August 2018; pp. 1059–1068. [CrossRef]
12. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; Volume 30.
13. Kang, W.-C.; McAuley, J. Self-attentive sequential recommendation. In Proceedings of the 2018 IEEE international Conference on Data Mining (ICDM), Singapore, 17–20 November 2018; pp. 197–206.

14. Sun, F.; Liu, J.; Wu, J.; Pei, C.; Lin, X.; Ou, W.; Jiang, P. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management, Beijing, China, 3–7 November 2019; pp. 1441–1450. [CrossRef]
15. Zhou, P.; Ye, Q.; Xie, Y.; Gao, J.; Wang, S.; Kim, J.B.; You, C.; Kim, S. Attention Calibration for Transformer-based Sequential Recommendation. In Proceedings of the 32nd ACM International Conference on Information and Knowledge Management, Birmingham, UK, 21–25 October 2023; pp. 3595–3605. [CrossRef]
16. Sun, Y.; Wang, B.; Sun, Z.; Yang, X. Does Every Data Instance Matter? Enhancing Sequential Recommendation by Eliminating Unreliable Data. In Proceedings of the IJCAI, Montreal, QC, Canada, 19–27 August 2021; pp. 1579–1585.
17. Wang, S.; Zhang, X.; Wang, Y.; Ricci, F. Trustworthy recommender systems. *ACM Trans. Intell. Syst. Technol.* **2022**. [CrossRef]
18. Fan, X.; Liu, Z.; Lian, J.; Zhao, W.X.; Xie, X.; Wen, J.-R. Lighter and better: Low-rank decomposed self-attention networks for next-item recommendation. In Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual, 11–15 July 2021; pp. 1733–1737.
19. Zhou, K.; Yu, H.; Zhao, W.X.; Wen, J.-R. Filter-enhanced MLP is all you need for sequential recommendation. In Proceedings of the ACM Web Conference 2022, Virtual, 25–29 April 2022; pp. 2388–2399.
20. Yuan, H.; Zhao, P.; Xian, X.; Liu, G.; Liu, Y.; Sheng, V.S.; Zhao, L. Sequential recommendation with probabilistic logical reasoning. *arXiv* **2023**, arXiv:2304.11383.
21. Rendle, S.; Freudenthaler, C.; Gantner, Z.; Schmidt-Thieme, L. BPR: Bayesian personalized ranking from implicit feedback. *arXiv* **2012**, arXiv:1205.2618.
22. Hidasi, B.; Karatzoglou, A.; Baltrunas, L.; Tikk, D. Session-based recommendations with recurrent neural networks. *arXiv* **2015**, arXiv:1511.06939.
23. Tang, J.; Wang, K. Personalized top-n sequential recommendation via convolutional sequence embedding. In Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, Del Rey, CA, USA, 5–9 February 2018; pp. 565–573.
24. Wu, S.; Tang, Y.; Zhu, Y.; Wang, L.; Xie, X.; Tan, T. Session-based recommendation with graph neural networks. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; pp. 346–353.
25. Wu, L.; Li, S.; Hsieh, C.-J.; Sharpnack, J. SSE-PT: Sequential recommendation via personalized transformer. In Proceedings of the 14th ACM Conference on Recommender Systems, Virtual, 22–26 September 2020; pp. 328–337.
26. He, Z.; Zhao, H.; Lin, Z.; Wang, Z.; Kale, A.; Mcauley, J. Locker: Locally Constrained Self-Attentive Sequential Recommendation. In Proceedings of the 30th ACM International Conference on Information & Knowledge Management, Virtual Event, 1–5 November 2021; pp. 3088–3092. [CrossRef]
27. Li, J.; Wang, Y.; McAuley, J. Time interval aware self-attention for sequential recommendation. In Proceedings of the 13th International Conference on Web Search and Data Mining, Online, 10–13 July 2020; pp. 322–330.
28. Staelin, D.H. Fast folding algorithm for detection of periodic pulse trains. *Proc. IEEE* **1969**, *57*, 724–725. [CrossRef]
29. Anderson, J.G.; Hough, S.E. A model for the shape of the Fourier amplitude spectrum of acceleration at high frequencies. *Bull. Seismol. Soc. Am.* **1984**, *74*, 1969–1993.
30. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26–30 June 2016; pp. 770–778.
31. Ba, J.L.; Kiros, J.R.; Hinton, G.E. Layer normalization. *arXiv* **2016**, arXiv:1607.06450.
32. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
33. Zhao, W.X.; Mu, S.; Hou, Y.; Lin, Z.; Chen, Y.; Pan, X.; Li, K.; Lu, Y.; Wang, H.; Tian, C. Recbole: Towards a unified, comprehensive and efficient framework for recommendation algorithms. In Proceedings of the 30th acm International Conference on Information & Knowledge Management, Virtual, 1–5 November 2021; pp. 4653–4664. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.